International Journal of Computer Trends and Technology (IJCTT) – volume 23 Number 4–May 2015

Advanced secured Authentication through Firewall in NSC

Ms. Savita Verma[1], Dr. Pushpender sarao[2]
[1]*M.Tech Scholar, Dept. of Computer Sci. & Engineering, Somany Institute of Tech. & Management Rewari Haryana , India*
[2]*Aassistant Professor, Dept. of Computer Sci. & Engineering, Somany Institute of Tech. & Management Rewari Haryana , India*

Abstract: ***The network Security is the hottest topic in the current research scenario. The information security is really threatened by obnoxious users. With increasing vulnerabilities, caused by port scan attacks, replay attacks and predominantly IP Spoofing, targeting services, the network behavior is getting malevolent. But there is a lack of any clear threat model. The authors have endeavored to consider this problem in order to improve the network security and enhance secure shell daemon protection. A mechanism, QKNOCK, improving upon the potentialities of technologies such as port knocking and SPA (Single Packet Authorization), using Firewall and Cryptography, has been proposed.***

Keywords : ***QKNOCK, SSH Daemon, Network Security, Port knock, IP Spoofing, Key-Exchange, Single Packet Authorization, Fwknop, AfterGlow, Gnuplot.***

I. INTRODUCTION

Security in communication is a crucial research area because of the complex technical nature involved in data transmission. The problem creeps in when the intention of the one of the users connecting with the network becomes bad .The increased vulnerabilities, cause replay attack dictionary attack, IP Spoofing and packet crafting etc. Security attack is an action which compromised the security of information owned by an organization. If a server runs on non-vulnerable software, a port scan is not a serious threat, but in case of non-patched and 0-day exploit software it becomes big threat. A popular method of protecting against such network attacks is the firewall, which simply blocks all connection attempts to "internal" network hosts from "external" ones. One class of proposed solutions to this problem is "port knocking" wherein a firewall is deployed to protect a server, but before allowing a client connection to a particular service (e.g ssh,imap,pop) , the client must transmit an authenticated knock. But the goal of a port knocking scheme to conceal the set of services running on a network host, through existing implementations have serious flaws. Port knocking schemes generally use the port number within the TCP or UDP header to transmit information from the client to the server, whereas its successor Single packet authorization (SPA) scheme uses messages to be sent over any IP protocol; not just those that provide a port over which data is communicated. Improving upon Fwknop (Firewall Knock Operator) currently supports sending SPA messages over ICMP or TCP. The technique Port Knocking and SPA has been used interchangeably. The above issues has been addressed and tried to be sorted out with modified enhanced approach.

In this paper, we have developed a formal security model QKNOCK which captures above notion. A formal security model is critically important in order to be certain that a given protocol, even one that seems secure at a glance, is secure in true sense. Examples of such "apparently secure" protocols, developed without formally stated security goals, are numerous and some of them have been in operation for years (and have even become industry standards). So much so all those protocols were originally designed for security, and even used well-known cryptographic primitives, but the protocols were not secure.

A. Formal Definition Of Single Packet Authorization (SPA)

SPA is a method of limiting access to server and network resources by cryptographically authenticating legitimate users before any type of TCP/IP stack access is allowed. The predominant researcher Michael Rash has given authenticated and effective solution to this security issue using fwknop. In the figure 1 below the client with IP address 14.4.4.4 through which an authenticated single packet has been sent to access services (daemons) that resides on the servers with IP address range 192.168.2.0/24. The firewall is reconfigured in such a way that only authenticated packet from legitimate IP address is received by the server in a default-drop stance.

II. SYSTEM DESIGN

In this section, the new modified scheme QKNOCK, has been introduced and implemented, displaying secure port knocking scheme, and discuss how this enhanced security model implementation averts number of limitations of previous systems which only attempt to authenticate the start of a connection, but provides no guarantee that connections stay authentic. In other words, previous implementations does not protect against attacks such as connection hijacking, IP spoofing, packet crafting, client/server synchronization, and indistinguishability. Next comes the analysis of number of possible attacks on these implementations. Finally, the results are graphically shown using AfterGlow. QKNOCK is designed to be an application-agnostic transport-level authentication layer. It prevents forgery and IP spoofing, packet crafting while hiding the presence of authentication scheme. The kernel hooks has been used to ensure that applications do not need to explicitly support the system in order to benefit from it. The keyed MACs are used as secure authenticators to stop forgery attacks and a two-part counter to counteract IP spoofing and packet crafting. These MAC's ensures that client and server counters stay synchronized even in the presence of packet loss. This is an implementation of a previously proposed operating system-specific steganographic embedding scheme for TCP/IP and use it to insert authentication information into TCP headers.

A. Universal Compatibility

This is a user friendly scheme by choosing an application agnostic design. By using hooks directly into the operating system kernel, avoids modifying any of the network kernel or library calls made by application software or requiring supports for SOCKS-type proxies. This allows any application to transparently use QKNOCK (without application awareness or modification), provided that the network protocol used by the kernel module part gives increased speed and efficiency over the user-space part . This approach allows any application to transparently use QKNOCK provided that the network protocol used by the application has a steganographic embedding/extraction method supported by QKNOCK. It is to be noted that for certain protocols, such as TCP, with many implementations which may have minute differences; each implementation may require a different steganographic embedding routine to preserve indistinguishability. The goal is to support as many transport protocol implementations as possible, although currently only TCP under RHEL 2.6.18.164.el5 is supported.

B. Design Choices

This proposed implementation is designed to run on the Linux operating system with a 2.6.18.164.el5 kernel. The choice of Red hat Linux 2.6.18.164.el5 is due to familiarity with the system and the availability of the net filter API.The encryption uses Poly1305- AES as the MAC function since it is optimized specifically for network packets and has very fast implementations available for various types of machines. Here Murdoch and Lewis' system for embedding steganographic information into TCP initial sequence numbers (ISNs) is used and the TCP timestamp option (enabled by

default in current Linux Kernel) to embed an additional byte of information into the timestamp, delaying packets when needed.

III. PROTOCOL USED

The QKNOCK Pseudo Code is outlined in Figure 1. This has addressed the vulnerabilities of previous Implementations. A QKNOCK client starts a connection which is composed of a TCP SYN packet to a QKNOCK-enabled server and steganographically embeds an authentication token into the packet. The embedding algorithm and resulting packet header structure has been described in Figures 3 and 4, respectively. The server receives a SYN packet and extracts the authenticator. If verification is successful, the server allows the connection to continue, otherwise the packet is dropped. The client and server share a key, as well as a counter which is incremented for every client connection attempt. The counter prevents IP spoofing, packet crafting by ensuring that every SYN packet sent by the client is different from any packets sent previously, and is also used as the nonce required by the MAC function. The key, initial counter and resynchronization interval are exchanged out of band, since negotiation is impossible in case of single-way communication.

A. MAC

A Message Authentication Code (MAC) is a short piece of information, similar to a hash code. It provides both original authentication and integrity protection of a message .The use of keyed MAC is preferred to additional series of knocks , applying it to the source and destination (IP, port) tuples as well as the counter, so every connection attempt is guaranteed to contain a unique MAC. The algorithm Poly1305-AES is deployed , for MAC function since it is designed specifically to work on small bits of data such as network packets and is implemented in optimized assembly for a number of popular platforms. The connection counter serves as the nonce required by Poly1305-AES. Considering that AES is a pseudorandom permutation, an attacker should not be able to compose a valid MAC, or even identify one from random bits, for the next SYN packet without knowing the key, keeping in view its visibility to outer world. MACs are generated by block ciphers and use symmetric secret keys to ensure that only those who know the key can modify, or verify the message.

B. Steganographic and Indistinguishability

The Pseudo Code for QKNOCK has been developed in figure 2.This method uses the TCP sequence number and timestamp fields of the TCP SYN packet to embed the MAC information. Due to limitation of embedding only 32 bits ,it is impossible to include the complete MAC, in current implementations. (24 bits in the sequence number and 8 bits — the least significant byte — in the timestamp), assuming Linux sequence number (see Figure 4). Since we must not allow distinguish ability based on discrepancy between the observed packet dispatch time and the packet timestamp, we delay packet transmission, but only use the last timestamp byte to minimize delay times. Although 32 bits is a relatively short MAC, keeping in view, an attacker can generate, on average, 232 packets to break the authentication (requiring, for example, 6 weeks to transmit over a T1 link). It is clear that standard methods to deal with online guessing attacks can also be applied here, such as account freezing or processing delays.

The Pseudo Code for QKNOCK is as under

1. Y→ X : Ny, MACkreq (Ny, Nx, ID X, req)
2. X→ Y : MACkreq (Nx, Ny, ID y, req)
3. Y → X : MACk, ctry(I) ; encoded in TCP/IP headers of SYN Packets
4. X : Set ctrx← ctrx + 1
   for i = 0 to ft
   if ((MACk, ctry – 1 + i (I) = MACk, ctr(I))
   Set ctrx ← ctrx + I + 1;
   Resynchronizes counter if client is ahead
   X→Y : SYN – ACK
   goto 7
5. Y : If (SYN – ACK received) then

Set ctry ← ctry + 1, goto 7;
Connection successful
6. Y : if ((SYN – ACK) not received) then
Set ctry ← ctry + 1;
Considering server recieves SYN,
but SYN – ACK was lost
goto 5
7. X, Y : proceed with TCP connection
if (FIN or RST received) then goto 1
Where,
X is server and
Y is client
req is a request for authentication,
Nx is a nonce chosen by X,
kreq is a secret key shared by X and Y,
IDx is the IP address of X,
ctrP is a per-IP- address counter maintained by principal P,
k is value derived from Y's IP address and a symmetric key shared between X and Y,
l is a TCP flow identifier,
ft is a failure-tolerance parameter,
MAC is a cryptographic message authentication function, represents concatenation

Fig. 1 The Pseudo Code for Proposed QKNOCK

In this scenario the main issue is concerned with the lost SYN packets. However, TCP requires that re-transmitted SYN packets have the same sequence number but different timestamp, so there is no encoding of stegotext in the timestamp: if the SYN packet was lost due to a malicious host, or if an attacker is observing all SYN packets, then the attacker would detect that the least significant byte of the timestamp in the original and re-transmitted SYN packets are same. The probability of this is only 1/256, so the attacker could guess existence of scheme.

S : TCP SYN packet
Sseq={P1,P2,P3,P4}:Sequence number of packet (4 bytes)
Stp ={T1,T2,T3,T4}:Timestamp of packet S (4 bytes)
l = (IPY,source port,IPX,destination port) : Authenticator
MACK,ctr (l) = {L1 , L2 , . . . , Ln} : n byte MAC
P2 = L1 , P3 = L2 , P4 = L3
T4 = hl ({T2 ||T3 }) : n-Universal hash function

Figure. 2.The steganographic encoding protocol. Decoding is done by reversing the operations in this protocol.

To sort out this issue, we ensure that the last byte of the timestamp be ambiguous to the attacker, even when trying to re-transmit the same MAC. As in Figure 2.

The current implementation use Murdoch and Lewis' system for embedding steganographic information into TCP initial sequence numbers (ISNs) and use the TCP timestamp option(enabled by default in Linux Kernel) to embed an additional byte of information into the timestamp, delaying packets when needed [14]

C. Counter Management Scenario

To protect against IP spoofing, packet crafting, per-user counter is employed, incrementing every connection attempt. If a given user has never before accessed a QKNOCK- protected server, the counter is started from 0 on both sides. The counter process gets disturbed when the client and server face desynchronization. Desynchronization is possible in two cases: either the client's SYN packet never reaches at the server, leading to the client

having a counter higher than the server's, or the server's SYN-ACK can be lost

Packet From Source Port	***Packet to Destination Port***
Adjusted for Internal Consistency	***Sequence No. MAC bytes 1-3***
Acknowledgement Number	
Offset Reserved	***Window Flag***
Checksum	***Urgent Pointer***
Timestamp Scenario	***Encoded MAC bytes 4***

Fig. 2 The TCP SYN packet after steganographic embedding.

Fig. 3 Synchronization from Server to Client

Fig.4 Synchronization from Client to Server

Further, it becomes difficult for the client to resynchronize connection if it fails to continue or whether the server received and verified the SYN packet but the SYN-

ACK was lost, or whether the SYN never arrived at the server in previous implementation. The permission is given by enhanced approach for an automatic in-protocol resynchronization after a certain time period. The figures 3 to 6 show the process of authentication and authorization of packets between client and server using cryptography and making use of the relationship.

Fig. 5 Encryption on Server Side

Fig .6 Public Key Authentications

the following equation has been used
ctrserver (qkknockd) ≤ ctrclient (knockSquid)

showing that by having the client always increment its counter when sending a SYN packet. The server, however, will only increment its counter upon successful MAC validation, to prevent malicious desynchronization by sending bogus packets to the server. In this scheme of insisting the counter to be exactly right, in sending the packet the server and client may never again get into sync once desynchronized, since the client will increment its counter on each connection attempt, but the server's counter remains the same. This scheme easily depicts the vulnerable access by malicious user.

IV. SYSTEM ARCHITECHTURE

The QKNOCK system is composed of two separate programs - "qkknockd" (server side), and " knock squid" (client side). Connections are authenticated on a per-flow instead of per-source (IP address) basis. The synchronization of packets at client is modified whereas on server side it remains unchanged. On both client and

server side, only sent packets are there into user space, to avoid excess context switching between user-space and kernel-space. Both qkknockd and knock squid currently detect closed connections as far as the Port knocking based potentialities are concerned.

A. Knock Daemon (Server)

QKNOCKED, is the server side daemon of the QKNOCK system, listens for connections on a port it reads from its configuration file (the port offering the protected service, i.e. SSH on port 22), and examines incoming SYN packets on those ports before the TCP/IP stack sees them. When a packet is received, qkknockd checks the source IP address of the packet and picks up the secret key as well as the counter for that IP address from its configuration file (per-user shared keys are also supported)[14]. Using the TCP steganographic algorithm, qkknockd extracts stegotext from the packet, treats it as a MAC, and attempts to verify it. If verification is successful the packet is accepted.

B. Knock Squid (Client)

Knock Squid reads a configuration file to find out which server supports QKNOCK, and for which services e.g IMAP, POP, SSH etc. The configuration file also includes the key shared with the server, and the last value of the connection counter (if this is the first time connecting to that server, the counter is started from 0). The number of initial firewall rules is linear in the number of QKNOCK-protected services which might be contacted.

In the architecture of QKNOCK the client communicates with server. The kernel creates a SYN Packet, Knock Squid receives packet before its departure embedding a MAC into the ISN and timestamp fields. Further, the server receives packet and its daemon gives it to the kernel. After successful picking up and verification of MAC by knock daemon, the packet is sent to application via kernel otherwise it is dropped. The qkknockd after receiving of SYN Packet does not examine other packets . However, knock squid rewrites every incoming and outgoing packet preventing client TCP Stack from getting disturbed due to sequence number mismatch.

V. SPA using PSAD and IPTABLES

Network Security is started with a properly configured firewall which allows basic network connectivity and services. In current technique, PSAD (Port Scan Detector) detects various types of suspicious traffic, such as port scans generated using Nmap and to address above issues fwknop [23] has been used.

A. Visualization of Malicious Packets using iptables Policy Configuration

Port scanning using NMAP is a malicious activity for interrogating remote targets. In the present work syslog data is supplied to AfterGlow which facilitates the process of generating graphs showing malicious data. In order to keep packets together, this script tells how to generate a graph (gif file) from a saved pcap (tcpdump) file ,reproduced below ,

tcpdump -vttttnnelr/home/rajesh/defcon.tcpdump | ./tcpdump2csv.pl "sip dip dport" | perl afterglow.pl -c color.properties | neato -Tgif -o test.gif

Calling afterglow.pl and specifying a color property file, this file is used by AfterGlow to determine the colors of the edges and nodes in the graph. Here we run Afterglow using following shell script Configuration File (Shell Script) Afterglow is shown by the color. Properties file that is used to configure the color output. As given below

```
# AfterGlow Color Property Fill
# @fields is the array containing the parsed values
# color.source is the color for source nodes
# color.event is the color for event nodes
# color.target is the color for target nodes
```

```
# The first match wins
color.source="yellow"
if ($fields[0]=~/^192\.168\..*/);
color.source="lightblue" if ($fields[0]=~/^10\..*/);
color.source="light blue"
if ($fields[0]=~/^172\.16\..*/);
color.source="red"
color.event="red" if ($fields[1]=~/^192\.168\..*/)
color.event="red" if ($fields[1]=~/^10\..*/)
color.event="red" if ($fields[1]=~/^172\.16\..*/)
color.event="red"
color.target="blue" if ($fields[2]<1024)
color.target="lightblue"
```

VI. CONCLUSIONS AND FUTURE RESEARCH

The schemes used in the existing PK/SPA implementation were flawed and vulnerable to attacks such as IP spoofing , packet crafting , connection hijacking , Ddos, No guarantee of backward and forward secrecy since the same keys were used. In addition to this, other issues which are such as inability to differentiate between port scans and port knocks, ignore IP address which have flooded firewall with bogus packets, code Complexities for embedded port knocking techniques using symmetric and asymmetric ciphers, limited packet loss while authenticating packets. we have designed a more secure SPA based authentication scheme QKNOCK which withstands some of these attacks. The current scheme is capable for recovery from packet loss while authenticating, differentiates port scans and port knocks. There is also an improvement avoiding code complexity in existing implementations for embedded port knocking techniques such as symmetric and asymmetric ciphers. However, current scheme does have issues which are to be addressed, such as the scheme is unable to ignore IP addresses that have flooded fwknop with bogus packets. There is also need to develop a benchmarking to find the current bottlenecks in the fwknopd server. Consequently, there is no current capability to generate large number of valid SPA packets in order to benchmark the fwknopd daemon. So, there is need to address above issues on wired and wireless communication networks.

REFERENCES

[1] Sebastian Janquier (2006) “Port Knocking Analysis with Single Packet Authorization” : Master’s Thesis, USA.

[2] http://www.linuxforu.com/2012/05/cyber-attacks-explained-packet-crafting/

[3] Isaacs, R., Jardetzky, P.,Mortier, R., Roscoe, T.: Techniques for lightweight concealment and authentication in IP networks. Technical Report IRB-TR-02-009, Intel Research Berkeley (July 2006).

[4] Rash Michael (2004) Available at Website http://www.cipherdyne.org/fwknop/docs/SPA.htm,

[5]http://www.cipherdyne.org/LinuxFirewalls/ch06

[6] Kumar Rajesh,Talwar I.M,Kumar Kapil , “ A Modified Approach to Analysis and Design of Port Knocking Technique”, International Journal of Computational Intelligence and Information Security(September 2012) Vol 3 (7), pp (28-39)

[7] Postel, J. (ed.): Transmission control protocol (1981),http://www.ietf.org/rfc/rfc0793.txt.

[8] Bleichenbacher,D.: Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS# 1.In :Krawczyk,H.(ed.)CRYPTO 1998LNCS,vol.1462,pp.1-12.Springer, Heidelberg(1998)

[9] Smits R., Jain D., Pidcock S., Goldberg I., Hengartner U. “Bridge SPA: Improving tor bridges with single packet authorization” (2011)Proceedings of the ACM Conference on Computer and Communications Security, pp. 93-101.

[10] M. Rash “Single Packet Authorization with fwknop” The USENIX Magazine,vol31,no1,Feb 2006.pp63-69[Online]Available http://www.usenix.org/publications/login/200602/pdfs/rash.pdf.

[11] Agrawal S., Boneh , D. Homomorphic MAC’s: MAC based integrity for network coding(2009) Applied Cryptography Network Security,5536,pp.292-305.

[12]http://www.cipherdyne.org/fwknop/docs/fwknop-tutorial.html#4.2 (October 2012)

[13] Eugene Y. Vasserman,Nicholas Hopper,John Laxson,and James Tyra "SILENTKNOCK : Practical, Provably Undetectable Authentication Vol.8, pp. 121- 135 (2009).Available at http://sclab.cs.umn.edu/node/151